An Introduction to Reinforcement Learning

Angela Carnevale

MathLearn - Galway, June 2026

0. Outline & References

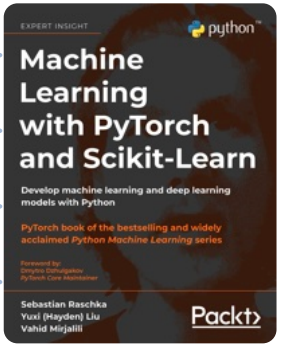
A. Intuition & basic concepts

B. Mathematical framework : Markov * processes

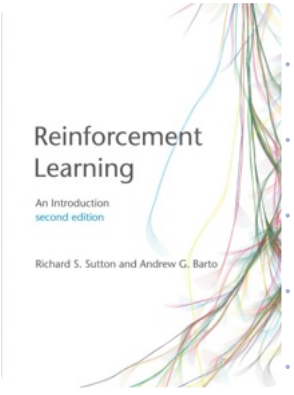
C. RL algorithms

0. Outline & References

* Machine Learning with PyTorch and Scikit-Learn
  Raschka, Liu, Mirjalili



* Reinforcement Learning: An introduction
  Sutton and Barto



* Reinforcement Learning (course notes)
  David Silver

Motivation & Basic Idea

RL as a "mathematical framework for learning to act".

1. Motivation & Basic Idea

Recall: Supervised learning: labelled data (x_i, y_i)

Goal: learn $x \longmapsto y$

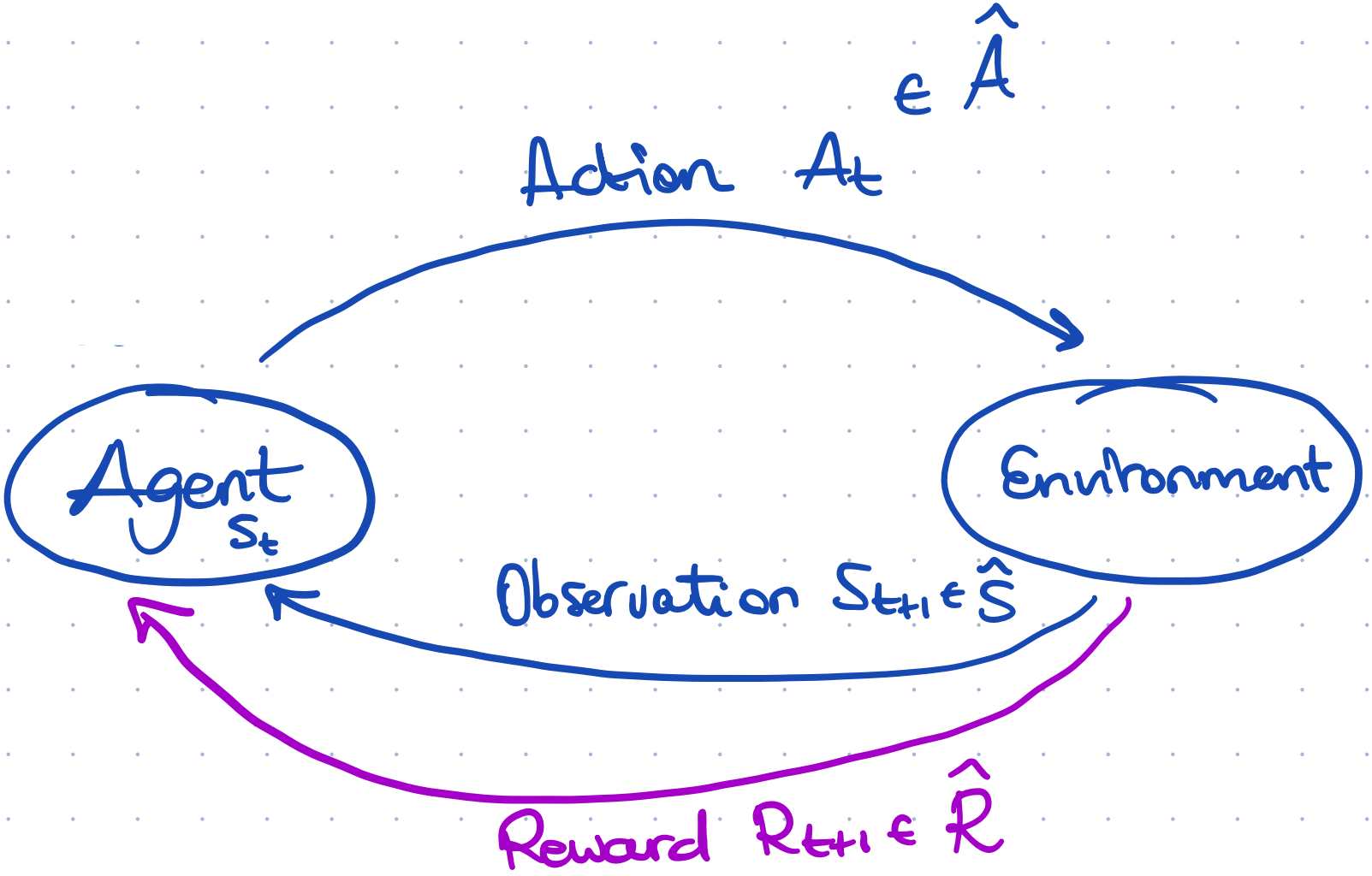
Here: learn to "act well" by interacting with an environment

Setup: Agent (= model) "decision entity"

Environment (= "World \ Agent") { reward + observation

Reward signal: feedback from environment to agent in response to interaction (a number!)

1. Motivation & Basic Idea

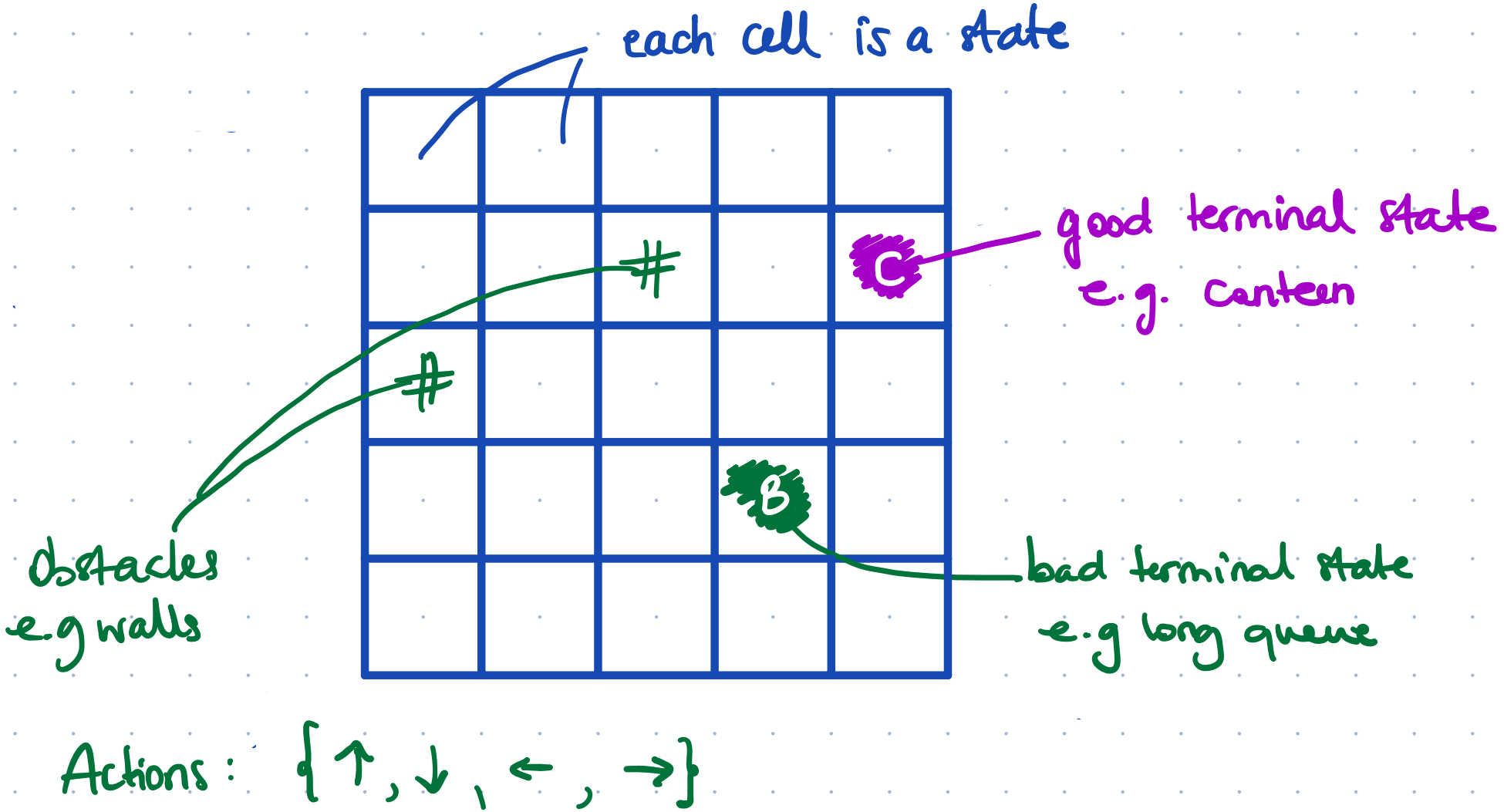


Slogan update:

RL is about learning good behaviour from reward-mediated interaction with an environment.

1. Motivation & Basic Idea

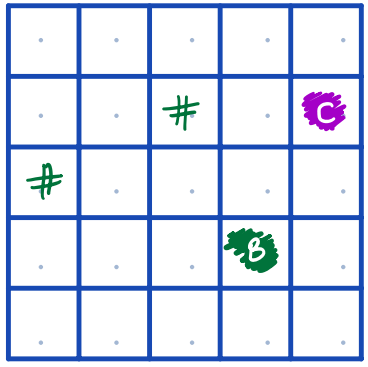
Example GridWorld



Actions: $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$

1. Motivation & Basic Idea

Example GridWorld



Q

How do we design rewards to make the agent learn "good behaviour"?

We need to Clarify what we actually want from the agent

Example • If our primary goal is to get to the canteen we might start by assigning a large positive reward to the state ● and a negative reward to the bad terminal state ●

• If we want to get to the canteen quickly, we might want to add a (smaller) negative reward to each remaining state.

But then why would the agent ever move?

2. Reward, Return, and Value

Key point: immediate rewards are not enough.

(think: take a detour to avoid a bad state to get to canteen, or simply do one step that gets us closer but not quite there!)

We will use the RETURN at time step t: some function of $R_{t+1}, R_{t+2} \ldots$ (future rewards) and of a discount factor γ.

The return depends therefore on the environment and on the policy of the agent (= rule that it follows to choose actions).

The expected value of the return will be a measure of the "goodness of a state".

Time to get formal!

Markov * Processes

3. Markov Processes

A Markov process consists of

- A set of states $\hat{S}$ together with transition probabilities

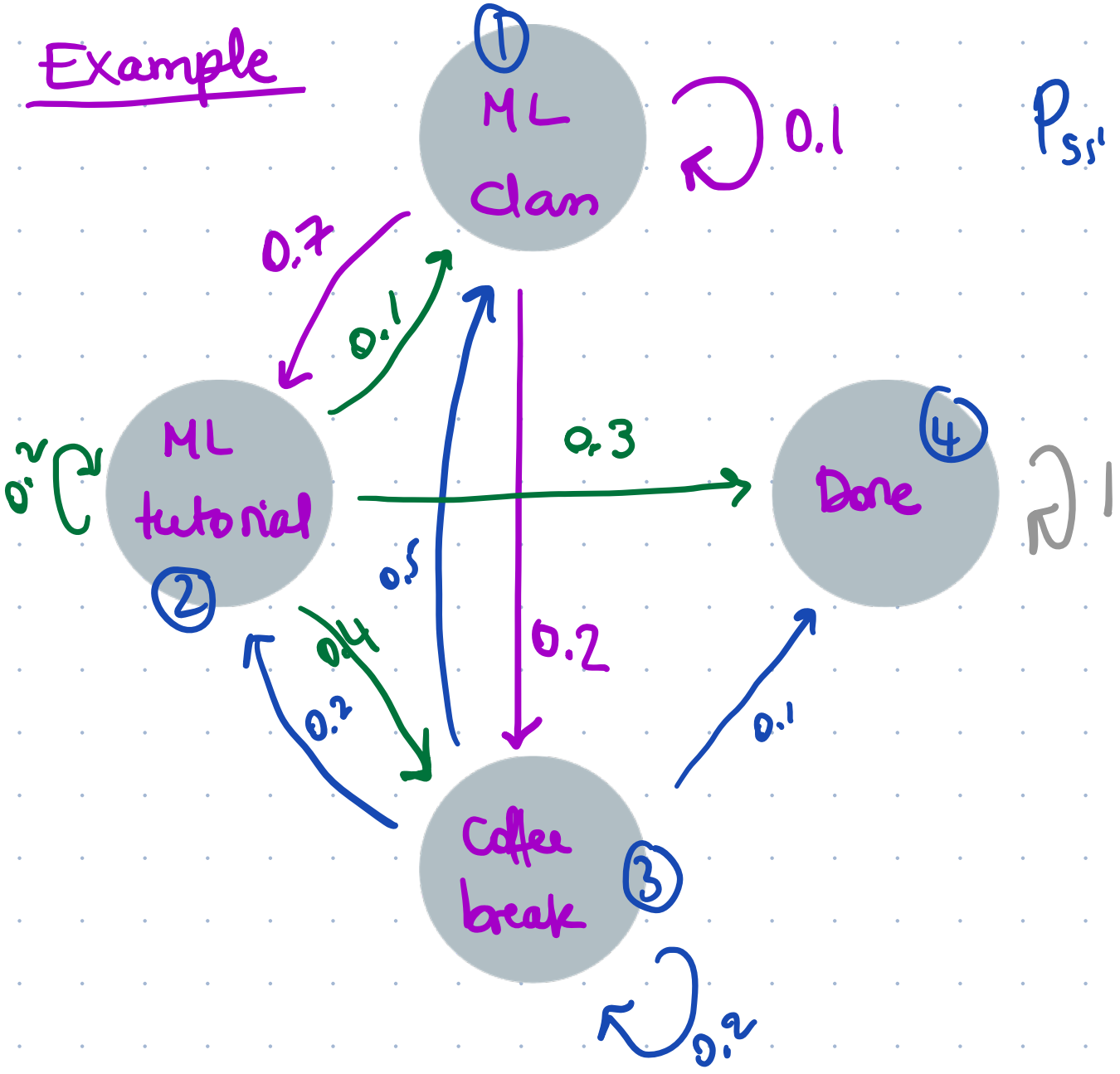
If $|\hat{S}| < \infty$, think of a matrix with entries

$$P_{ss'} := \underbrace{P(S_{t+1} = s' \mid S_t = s)}_{\text{MARKOV PROPERTY}}$$

$$P(S_{t+1} = s' \mid S_t = s, S_{t-1} = \tilde{s} \dots)$$

3. Markov Processes

Example



$$P_{ss'} = \begin{bmatrix} 0.1 & 0.7 & 0.2 & 0 \\ 0.1 & 0.2 & 0.4 & 0.3 \\ 0.5 & 0.2 & 0.2 & 0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Markov Reward Processes (MRPs)

An MRP consists of:

- A set of states $\hat{S}$
- A set of rewards $\hat{R}$

together with transition probabilities

- A discount factor $\gamma \in [0,1]$

States S_t, Rewards R_{t+1} are time-dependent random variables that only depend on their values at the preceding time step.

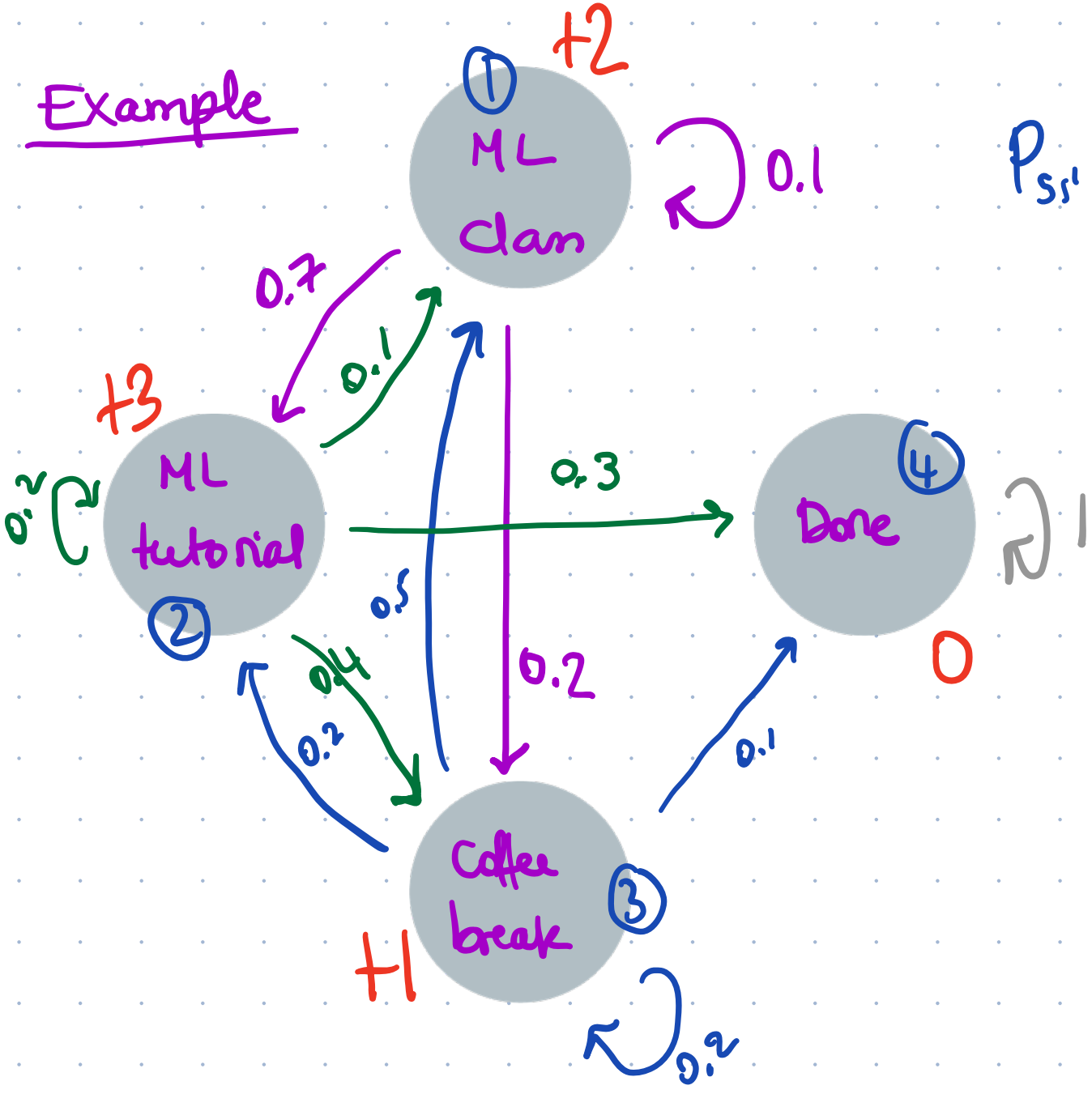
The probabilities

$$p(s',r|s) := \mathbb{P}(S_{t+1}=s', R_{t+1}=r \mid S_t = s)$$

completely determine the dynamics of the environment.

3. Markov Reward Processes

Example



$$P_{ss'} = \begin{bmatrix} 0.1 & 0.7 & 0.2 & 0 \\ 0.1 & 0.2 & 0.4 & 0.3 \\ 0.5 & 0.2 & 0.2 & 0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. MRPs – Return

This is what we will want to maximise!

The return at time t is

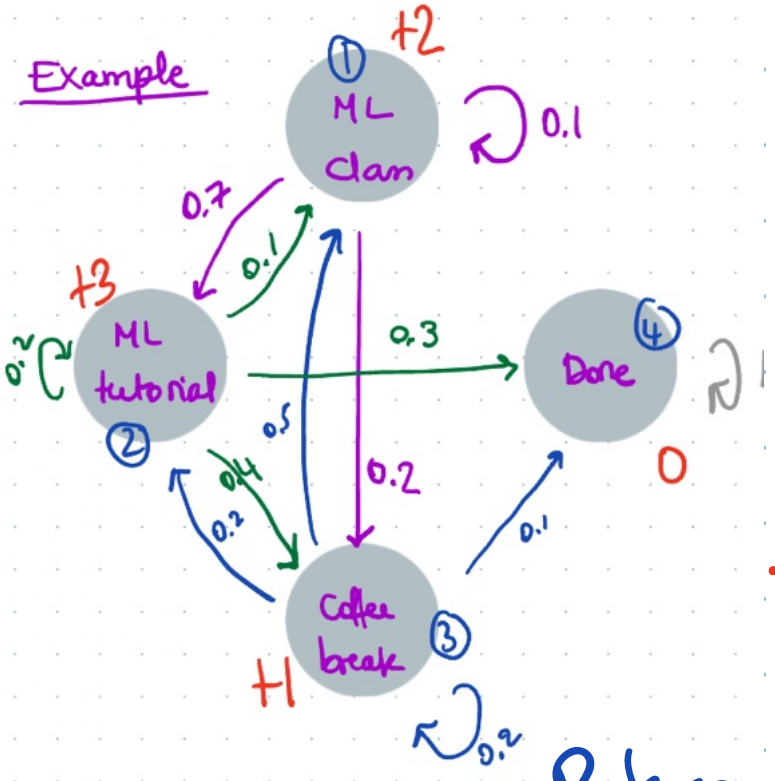
$$G_t := R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k \geq 0} \gamma^k R_{t+k+1}$$

- $\gamma = 0$: short-sighted agent
- $\gamma = 1$: return is unweighted sum of all future rewards

Note the following recursive expression for the return at time t:

$$\boxed{G_t = R_{t+1} + \gamma G_{t+1}}$$

3. Markov Reward Processes



$\gamma = 0.8$

Sample rewards for student MRP doing

* C-T-B-T-D

Return = $2 + 0.8 \cdot 3 + 0.8^2 \cdot 1 + 0.8^3 \cdot 2 = 6.06$

* C-C-C-B-B-D

Return = $2 + 0.8 \cdot 2 + 0.8^2 \cdot 2 + 0.8^3 \cdot 1 + 0.8^4 \cdot 1 + 0 = 5.$

3. MRPs – Value function

Value function (or state-value function) measures goodness of a state s, based on return G_t

$$V(s) = \mathbb{E}[G_t \mid S_t = s]$$
$$= \mathbb{E}[\Sigma_{k\geq 0} \gamma^k R_{t+k+1} \mid S_t = s]$$

Idea: conditioning to next state and passing to matrix form ($|S| < \infty$) get a Bellman equation

$$\underline{V} = \underline{r} + \gamma P \underline{V} \rightsquigarrow \underline{V} \text{ is sdh to linear system.}$$

Where's the agency?

Enter Actions:

Markov Decision Processes

4. Markov Decision Processes MDPs

An MDP consists of:

- A set of states $\hat{S}$
- A set of rewards $\hat{R}$
- A set of actions $\hat{A}$
- A discount factor $\gamma \in [0,1]$

together with transition probabilities

States S_t, actions A_t, Rewards R_{t+1} are time-dependent random variables that only depend on their values at the preceding time step.

The probabilities

$$p(s', r \mid s, a) := \mathbb{P}(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

completely determine the dynamics of the environment.

4. MDPs – Policy

A policy π is a function that determines the next action to take:

$$\pi(a|s) = P(A_t = a \mid S_t = s)$$

(or simply $\pi(s) = a$ if policy is deterministic)

Goal: Learn an optimal policy $\pi_*(a|s)$

(= policy that yields highest returns).

4. MDPs – Value function

Value function (or state-value function) measures goodness of a state s, based on return G_t and assuming we're following policy π:

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

$$= \mathbb{E}_\pi\left[\sum_{k \geq 0} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

"How good is it to be in state s under policy π"?

4. MDPs — State-action function

If we look at state-action pairs, we get

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

$$= \mathbb{E}_\pi\left[\sum_{k\geq 0} \gamma^k R_{t+k+1} \mid S_t = s, A_t = A\right]$$

"How good is it to take action a (from s) and then continue to follow policy π"?

4. MDPs – Value & State-action function

V_π and q_π are related via:

$$V_\pi(s) = \sum_{a \in \hat{A}} \pi(a|s)\, q_\pi(s,a)$$

knowledge of $q_\pi(s,a)$ means we know immediately what to do next:

Choose a such that $q_\pi(s,a)$ is largest.

this is the "Q" of Q-learning and DQN.

4. MDPs – Bellman expectation equation

Back to the def of v_π:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

expand + condition on next state + linearity of expected value

$$\rightsquigarrow \quad v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s) \cdot [r + \gamma v_\pi(s')]$$

again: can solve for $v_\pi = (I - \gamma P^\pi)^{-1} r^\pi$

We're not optimising anything, yet!

5. Policy improvement & optimality

Given π and V_π, our goal is to improve the policy, i.e find π' such that

$$V_{\pi'}(s) \geq V_\pi(s) \quad \forall s \in \hat{S}$$

Idea: improve policy by computing $q_\pi(s,a)$ for all pairs s,a, then choose an improved policy π' to have

$$\pi'(s) \in \operatorname{argmax}_a q_\pi(s,a)$$

5. Policy improvement & optimality

Fact. For any MDP

- There exists an optimal policy π_*
- All optimal policies achieve the optimal value function $v_* \left(= \max_\pi v_\pi\right)$
- All optimal policies achieve the optimal action-value function $q_* \left(= \max_\pi q_\pi\right)$

5. Policy improvement & optimality

An optimal policy can be found by maximising over $q_*(s,a)$:

$$\pi_*(s|a) = \begin{cases} 1 & \text{if } a \in \underset{a}{\operatorname{argmax}}\, q_*(s,a) \\ 0 & \text{o/w} \end{cases}$$

$\rightsquigarrow$ get a deterministic optimal policy

Teaser: E vs E

5. Bellman optimality equations

However, now the Bellman eqn become non linear:

$$V_*(s) = \max_a q_*(s,a)$$

$$= \max_a \sum_{s'} p(s'|s,a)\left(r(s,a,s') + \gamma V_*(s')\right)$$

These equations have no closed form solution in general

Enter RL

Q-Learning & other iterative methods

Before that: should our agent ALWAYS take the action that always look best?

6. Exploration and Exploitation

Bellman optimality tells us what V_* must satisfy.

But what if the agent does not know all environment dynamics?

"ε-greedy policy", mix of

Exploitation: follow current known best

Exploration: take random action (= explore)

$$\pi(a|s) = \begin{cases} \text{best known action w/prob. } 1-\varepsilon \\ \text{random non-optimal action w/prob } \varepsilon \end{cases}$$

Working on a "need-to-know" basis:

Model-based vs Model-free

Goal: learn how to act best, not learn environment dynamics

7. Model-based vs Model-free

In a **Model-based** scenario:

agent learns all env dynamics, plans according to model ($p(s'|s,a), r(s,a,s')$)

In a **Model-free** scenario:

agent only learns values or policies directly (not transition probabilities) and learns to act from sampled experience

↓

TD, Q-Learning, DQN, PPO..

Towards Q-Learning:

TD prediction & Control

Recall: we need some way to approximate V and Q (and learn an optimal policy)

8. TD = temporal difference prediction

compute / estimate V_π

We use this method to evaluate a fixed policy π from experience.

Suppose we know a current estimate $V(S_t)$ and we observe the sequence

$$S_t \overset{s}{\rightarrow} R_{t+1} \overset{\pi}{\rightarrow} S_{t+1} \overset{s'}{}$$

We can use this observation to improve our current estimate of $V(S_t)$ based on this sample. The TD error is

$$G_{t,t+1} - V(S_t)$$

1-step "return" actually observed

8. TD prediction

We can then update our estimate of $V(S_t)$ according to the rule:

$$V(S_t) \leftarrow \underbrace{V(S_t) + \alpha \left(G_{t:t+1} - V(S_t) \right)}_{\text{TD target}}$$

$$V(S_t) + \alpha \left(\overbrace{r + \gamma V(S_{t+1})}^{\text{TD target}} - V(S_t) \right)$$

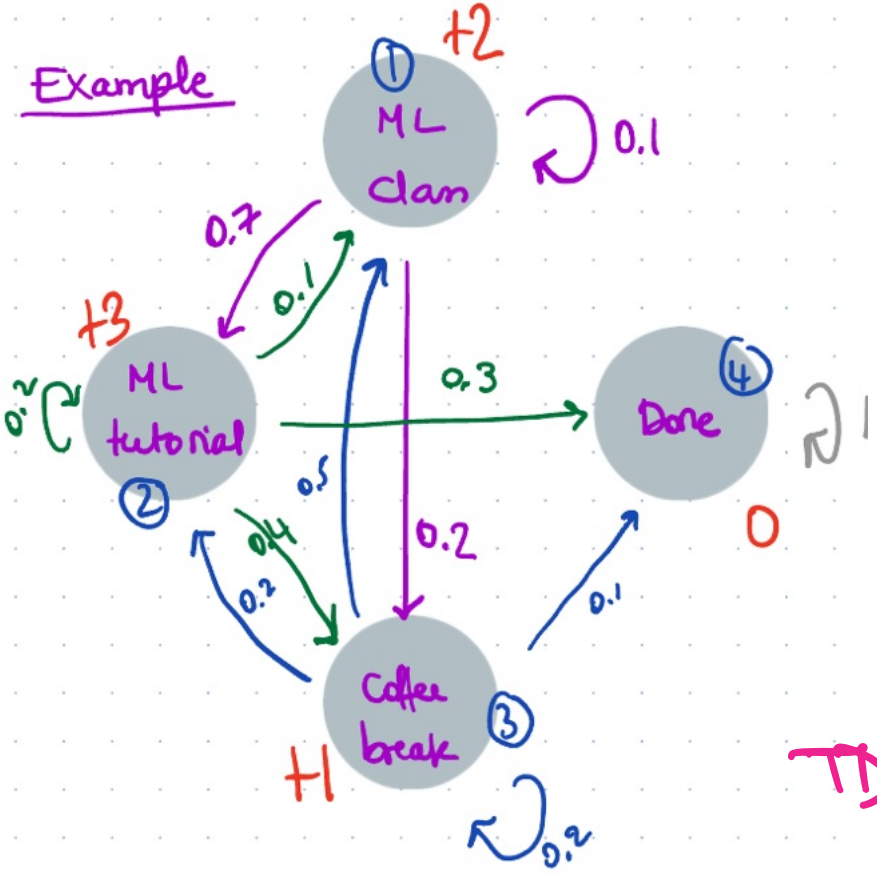
where α is a hyperparameter called learning rate.

Note: we can use $V(s) = 0 \; \forall s \in \hat{S}$ as initial estimate, or some random initial value [always 0 for terminal states]

8. TD prediction

update rule $V(S_t) \leftarrow V(S_t) + \alpha \left(r + \gamma V(S_{t+1}) - V(S_t) \right)$

Example



Suppose our current estimate is

$V = [4, 6, 3, 0]$; $\gamma = 0.9$, $\alpha = 0.1$

+2

and we observe Class → Tutorial

TD target $2 + \gamma \cdot 6 = 2 + 5.4 = 7.4$

TD Error is $7.4 - 4 = 3.4$

Update: $V(\text{Class}) \leftarrow 4 + 0.1 \cdot 3.4 = 4.34$

new $V = [4.34, 6, 3, 0]$

9. TD control

We managed to update our estimate of V following a policy π. We now want to learn from this.

Whether we learn to improve the current policy, or to learn a target policy is the difference between

on-policy: we can imagine behaving according to a policy and learning to improve the same

off-policy: we behave according to a policy, we learn what's best and we make that into another policy.

9. TD control

Either way, to learn we need to go back to the state-action function (or better, its estimate) Q.

Again suppose we observe $S_t, A_t, R_{t+1}, S_{t+1}$

On-policy update: SARSA (we also need A_{t+1})

target: $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$

update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right)$$

We learn from observation, including exploration

Q-Learning:

hindsight for future action

10. Off-policy: Q-Learning

Here, we only need $S_t, A_t, R_{t+1}, S_{t+1}$

and we use as target the best estimated next action

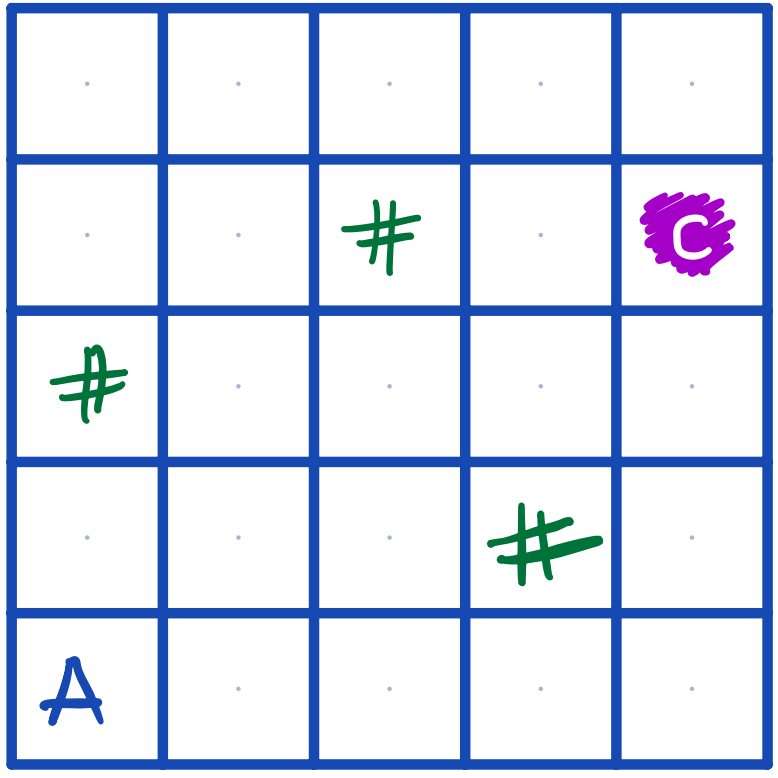
target: $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$

update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right)$$

We might still be exploring, but we learn as if we will act greedily in the future.

Back to GridWorld



State Reward

G goal +10

walls −2

each step −1

Actions: {↑, ↓, ←, →}